DataGRID

USER GUIDE FOR EDG REPLICA MANAGER 1.4.3

EDG REPLICA MANAGER

Document identifier:	**DataGrid-02-ERM-USER-V1.0**
EDMS id:	**http://edms.cern.ch/document/375434/**
Date:	August 26, 2003
Work package:	**WP2**
Partner(s):	**CERN**
Lead Partner:	**EDG**
Document status:	**Version 1_1_0**
Author(s):	James Casey, Peter Kunszt, Erwin Laure, Heinz Stockinger, Kurt Stockinger
File:	**edg-replica-manager-userguide**

Abstract: This document gives an overview of how to use the EDG Replica Manager.

CONTENTS

1. INTRODUCTION **4**
1.1. GENERAL HINTS FOR USAGE OF THIS DOCUMENT 4
1.2. GLOSSARY 5
1.2.1. NAMING CONVENTIONS 5
1.2.2. DATA MANAGEMENT 6
1.2.3. REPLICA MANAGEMENT 6
1.3. DATA NAMING 6

2. USER INTERFACE **8**
2.1. DIFFERENCES TO THE PREVIOUS `EDG REPLICA MANAGER` 8
2.2. COMMAND LINE INTERFACE 8

3. COMMANDS **10**
3.1. MANAGEMENT COMMANDS 10
3.1.1. `COPYANDREGISTERFILE` 10
3.1.2. `REPLICATEFILE` 12
3.1.3. `DELETEFILE` 14
3.2. CATALOG COMMANDS 15
3.2.1. `REGISTERFILE` 16
3.2.2. `REGISTERGUID` 17
3.2.3. `UNREGISTERFILE` 18
3.2.4. `LISTREPLICAS` 19
3.2.5. `LISTGUID` 20
3.2.6. `ADDALIAS` 21
3.2.7. `REMOVEALIAS` 22
3.2.8. `PRINTINFO` 23
3.2.9. `GETVERSION` 24
3.3. OPTIMIZATION COMMANDS 24
3.3.1. `LISTBESTFILE` 24
3.3.2. `GETBESTFILE` 25
3.3.3. `GETACCESSCOST` 27
3.4. FILE TRANSFER COMMANDS 28
3.4.1. COPYFILE 28

4. QUICK START AND SIMPLE HOW TO **29**
4.1. FIRST STEPS 30
4.2. REPLICATION EXAMPLES USING OPTIMIZATION FUNCTIONS 30

5. APPENDIX: EDG REPLICA MANAGER LOG FILE **34**

6. APPENDIX: REPTOR VERSUS `EDG-REPLICA-MANAGER` 35

CONTENTS

Document Change Record

Issue	Date	Comment	Author
0_1	4/03/03	Expanded with this DCR, Glossary and reworked the Intro. Previous version by Kurt Stockinger.	Peter Kunszt
0_2	25/03/03	Section on EDG Replica Manager vs Reptor added, some small fixes, new option -vo added	Heinz & Kurt Stockinger
1_0	26/03/03	Options –vo and –config described	Peter Kunszt
1_0_1	28/03/03	Description of the log file usage added	Peter Kunszt
1_0_2	28/04/03	Description of few new commands added	Kurt Stockinger
1_0_3	5/06/03	Minor editorial changes	Heinz Stockinger
1_0_4	12/06/03	Examples for file names added already in the beginning of the document. Several editorial changes, additions and corrections of commands.	Heinz Stockinger
1_1_0	16/06/03	Changes for filenaming - we now use SURL instead of SFN with the following syntax srm://hostname/path/filename	Heinz Stockinger

1. INTRODUCTION

The data management services of WP2 are a set of coordinated high-level grid middleware services. The set of services dealing with file cataloguing, replication and copying are described in [1]. We call this replica management framework 'Reptor'. The EDG Replica Manager (ERM) coordinates all of the Reptor services and is the Grid user's entry point to them.

We will not go into the details of the Reptor service design in this document, the reader is referred elsewhere for details [1, 2, 7].

In this document we focus on the functionality and usage of the EDG Replica Manager.

1.1. GENERAL HINTS FOR USAGE OF THIS DOCUMENT

In order to use this document in an optimal way, we recommend you to go through the document as follows:

- The **Glossary** Section 1.2. gives you all the necessary background information on terminology and file naming. You will also find some examples there. The section is rather useful for beginners.
- In a similar way, also Section 1.3. provides useful background information on file cataloging.
- Section 2. **User Interface** is very important if you use the tool the first time since it gives all basic information of how the command line interface works. In addition, a simple **Quick Start Guide** and **How To** is provided in Section 4.. We recommend that you read both of these sections carefully.

  Details for each of the commands (with all possible arguments, options etc.) provided by the EDG Replica manager are given in Section 3. **Commands**.
- For people already familiar with the previous Replica Manager provided by WP2 in EDG release 1.4.x (called `edg-replica-manager`, some hints are given in the Appendix in Section 6. in order to provide an easier transition to this new tool described in this user guide.

1.2. GLOSSARY

1.2.1. NAMING CONVENTIONS

SURL	**Storage URL**	An SURL is a locator for a physical file, where the scheme specific part is understood by a Storage Resource Manager (SRM). It is a URL where the scheme is 'srm' and the host is a valid SRM host.
UUID	**Universally Unique IDentifier**	A UUID is a 128 bits long number, and is either guaranteed to be different from all other UUIDs generated until 3400 A.D. or extremely likely to be different (depending on the mechanism chosen to generate it).
GUID	**Grid Unique IDentifier**	A UUID generated by the Replica Management System for an SURL. It is created at the SURL registration time. A GUID is immutable.
LFN	**Logical File Name**	A Logical File Name is a user defined alias to a GUID. Unlike GUIDs, aliases are mutable but they still should be globally unique. Since the Replica Management System has no control over the creation of LFNs, this global uniqueness is only weakly enforced.
TURL	**Transport URL**	A Transport URL is returned by a SRM in response to a request for a way to access a SURL. It includes the actual protocol you can access the SURL by. For instance, 'gsiftp' for GridFTP, or 'http' for HTTP access.

EXAMPLES

In the following we give examples for the definitions of naming conventions.

SURL examples:

```
srm://host1.cern.ch/directory1/directory2/filename
srm://lxhare0384.cern.ch/flatfiles/cms/data/05/x.dat
```

GUID examples:

```
guid:73e16e74-26b0-11d7-b1e0-c5c68d88236a
guid:7c29f32b-4964-11d7-a86c-9ee9a33b1f19
```

LFN examples:

```
lfn:mydata
lfn://any_name_you_want
```

TURL examples:

```
gsiftp://host1.cern.ch/directory1/directory2/filename
http://lxhare0384.cern.ch/flatfiles/cms/data/05/x.dat
```

1.2.2. DATA MANAGEMENT

DMS	**Data Management Services**	The collection of all services for Grid data management, including file I/O, mass storage, access and replication. WP2 provides the Replica Management Services for the DMS.
RMS	**Replica Managemement Services**	The collection of all replication services provided by WP2: Replica Location Service, Replica Optimisation Service, Replica Metadata Catalog and Replica Storage Handler. The EDG Replica Manager provides a consistent command line interface to all of these components.
RFT	**Reliable File Transfer**	A service that transfers data reliably over the wide area network.
SRM	**Storage Resource Manager**	A high-level interface to a storage system.

1.2.3. REPLICA MANAGEMENT

ERM	**EDG Replica Manager**	The client interface to all RMS operations. This consists of a set of command-line tools.
RLS	**Replica Location Service**	The distributed service providing the mappings between GUIDs and SURLs. An RLS has two components: Local Replica Catalogs and Replica Location Indexes.
LRC	**Local Replica Catalog**	The catalog storing GUID to SURL mappings, along with SURL attributes for a given site, or a single Storage Resource Manager at a site. It only stores GUID to SURL mappings for SURLs that are actually located in the given site or SRM.
RLI	**Replica Location Index**	The catalog storing information about which Local Replica Catalogs have GUID to SURL mappings for a particular GUID. It thus provides the link between different LRCs, allowing for distributed indexing and querying of the Catalogs.
RMC	**Replica Metadata Catalog**	The catalog storing LFN aliases for GUID, as well as attributes on GUIDs and LFNs.
ROS	**Replica Optimization Service**	A service providing information to guide selection between replicas located at different sites. This is based on network information collected from available network monitors.
RSS	**Replica Subscription Service**	Middleware managing automatic replication between Storage Resource Managers based on subscription information.

1.3. DATA NAMING

EDG Replica Manager manages data held in files. In a Grid environment the data files are replicated, possibly on a temporary basis, to many different sites depending on where the data is needed. The users or applications do not need to know where the data is located. Having a logical name for the data file they should be able to use the data management services to locate and access the data.

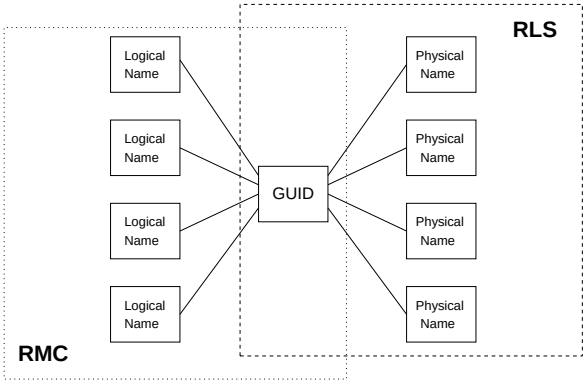

Figure 1: The Logical File Name to GUID mapping is maintained in the Replica Metadata Catalog, the GUID to Physical File Name mapping in the RLS.

Users and applications can use the Grid Unique IDentifier (GUID) to locate the data; these are assigned at data registration time and are based on the UUID standard [**?**] to guarantee unique IDs. GUIDs are immutable. All the replicas of the data will share the same GUID.

In order to locate a Grid accessible file, the human user will normally use a Logical File Name (LFN). LFNs are usually more intuitive, human-readable strings as opposed to the cryptic GUIDs, and are allocated by the user. Within the Grid LFNs must be globally unique as well, each LFN being an alias for exactly one GUID. The logical name space can be partitioned at will by the users. A GUID may have many LFNs as aliases. In this document we also call LFNs 'GUID aliases'.

Given either a LFN or GUID, the Metadata and Replica Location Catalogs are contacted to obtain a Storage URL (SURL) for the physical instance(s) of the required file. The SURL is a URI having the Storage Resource Manager as its hostname which takes responsibility for the storage of the physical file. This Storage Resource Manager (SRM) should be able to accept the SURL and return a valid URL, a Transport URL (TURL) in our terminology, to the application. This TURL provides all the information required (i.e. protocol, host, port and path) so that the application can open and retrieve the required physical file.

Note that Reptor requires both catalogs, Replica Location Service and Replica Metadata Catalog, but an end-user should not need to distinguish between the two of them. In the remainder of the document we use the terms **Replica Catalog** or **Catalog** to refer to both of them regardless of their exact functionality.

2. USER INTERFACE

We offer command line interfaces (CLIs) and application program interfaces (APIs) to the ERM. In this user-guide we only discuss the CLIs and their functionality, the APIs (that are very similar) are described in the developer's guide.

2.1. DIFFERENCES TO THE PREVIOUS `EDG Replica Manager`

The command line executable, `edg-replica-manager` or `edg-rm` is by design very similar to its predecessor (edg-replica-manager [5]) with the same name on the previous release of the Testbed. The new ERM has more functionality and the arguments are usually parsed by placement, not by options as before. Most of the command names have been preserved but the arguments have changed so read this guide carefully if you are migrating your jobs and scripts. The detailed differences are in the appendix.

2.2. COMMAND LINE INTERFACE

The command line interface to the ERM can be accessed via the `edg-replica-manager` command, or `edg-rm` for short. It takes a set of global options and then a command that has in turn options of its own.

The global options are listed if the command is given without any arguments:

```
usage: edg-replica-manager [options] command [options]
 -h,--help            print help (if command is given, details on command)
 -i,--insecure        Connect in an insecure manner, i.e. not https.
    --config=<file>   read configuration from specified file
    --vo=<VO>         set Virtual Organization
 -v,--verbose         print additional information while executing
```

> Note that currently the VO needs to be passed to the replica manager, e.g. --vo=cms. The vo option will not be necessary once VOMS is available.
>
> Note also that due to a problem in argument passing, this option as well as the config option has to be specified using the '=' sign with no spaces.

The command can be any of the following, including the short-hand forms (the same description is printed using the -h option):

Management commands		
copyAndRegisterFile	cr	Put a local file into Grid Storage and register it in the Catalog
replicateFile	rep	Replicate an existing file to a certain Grid Storage, update Catalog
deleteFile	del	Delete a file from Storage and remove entry from Catalog
Catalog commands		
registerFile	rf	Register a file in the catalog
registerGUID	rg	Register a file with a known GUID in the catalog
unregisterFile	uf	Unregister a file from the catalog
listReplicas	lr	List all replicas of a logical file name
listGUID	lg	List the GUID of a known LFN or SURL
addAlias	add	Add an LFN alias to an existing GUID
removeAlias	ra	Remove an alias to GUID mapping
printInfo	pi	Print all info service data to screen or to a file that can be reused by the static configuration
getVersion		Get the replica manager version
Optimization commands		
getBestFile	gbf	Replicate a file to the 'nearest' Storage Element in the 'cheapest' way
listBestFile	lbf	List the replica that has the smallest access cost
getAccessCost	ac	List access costs for all replicas
File transfer commands		
copyFile	cp	Copy a file. Use this to copy a file to local store by specifying an LFN as the source and a local destination.
Deprecated commands – these will disappear with the next release		
registerEntry	re	Same as registerFile.
unregisterEntry	ue	Same as unregisterFile.

In the following sections these commands are described in detail.

Note: By default, the command line tools do *not* produce any log file, i.e. error logging is switched off. Error logging can be helpful for debugging reasons or if you want to get additional information on the error messages that are printed on the screen. Please refer to Section 5. for detailed information on how to switch on logging into a log file.

3. COMMANDS

3.1. MANAGEMENT COMMANDS

The management commands are:

- `copyAndRegisterFile`
- `replicateFile`
- `deleteFile`

3.1.1. `copyAndRegisterFile`

Command:	**copyAndRegisterFile**
Description:	Put a (local) file into Grid Storage and register it in the Catalog
Arguments:	**sourceFileName** The file to copy and register. It should be a fully qualified URI. Possible schemes are: file, gsiftp, http, https, ftp.
Options:	**-d** or **- -destination** *destination-SURL* or *SE host* **-l** or **- -logical-file-name** *logical-file-name* **-p** or **- -protocol** *protocol* **-n** or **- -streams** *number of streams*
Example:	`edg-replica-manager --insecure --vo=wpsix copyAndRegisterFile file:/home/bob/analysis/data5.dat -d lxshare0384.cern.ch`

The help printed if the command is invoked without arguments or using `edg-rm -h copyAndRegisterFile`:

```
usage: edg-replica-manager copyAndRegisterFile sourceFileName [options]
Copy a file to grid-aware storage and register it in the grid catalog. Upon
successful completion this call returns the GUID of this file that can be
used to retrieve it again.
 -d,--destination-file <file>    The destination file name (SURL) or
                                 destination SE host. If it is not given,
                                 the closeSE is found from the info
                                 services and an automatic file name is
                                 used as an SURL.
 -l,--logical-file-name <file>   The logical file name to register this
                                 file with. Has to start with 'lfn:'. If it
                                 is not given, the only way to find a file
                                 is through the GUID that is returned by
                                 this call.
 -n,--streams <#streams>         The number of parallel streams to use.
                                 Defaults to 8. This only has an effect if
                                 the protocol supports parallel streams.
                                 Currently the only protocol to do so is
                                 gsiftp.
 -p,--protocol <protocol>        The protocol to be used for transfer.
                                 Defaults to gsiftp.
 Arguments:
  sourceFileName         The file to copy and register. It should be a fully
                         qualified URI. Possible schemes are: file, gsiftp,
                         http, https, ftp.
```

The `copyAndRegisterFile` call performs the task of copying a file into grid-aware storage and registering the copy in the Replica Catalog as an atomic operation. This is the preferred method to 'bring files into the grid'.

The same could be done by issuing `copyFile` and `registerFile` in sequence, but since `copyFile` may not have an SE or SURL as its destination, the user would need to specify a valid transport file name (using gridftp for example) to a location that happens to be the proper location for the file and then register it using `registerFile`. That is much more error prone and cumbersome than this method.

As a rule, `copyFile` should be used to copy files around to non-grid aware storage, or to simply copy files *out* of the grid (i.e. the source may be grid-aware but not the destination).

`registerFile` should be used to register files that are already in the grid-aware store before it was made grid-aware or to register files that appear there through other means (like new data files). If the files that need to be put into the grid are not yet in their grid-aware location, `copyAndRegisterFile` is the most robust method to use.

Semantics

`copyAndRegisterFile` also handles all the possible failures if the registration should fail after the copy has taken place (i.e. the copied file is registered using another name or if that is not possible it is actually removed again). If the destination already exists, the operation fails.

Re-registrations of existing data files are possible by issuing `copyFile` and `registerFile` in sequence as specified above, or by copying the file 'out' using `copyFile` and re-registering it again using this method but storing it with a different name. If users really want to do a re-registration resulting in having more than one GUID for the same data (of course not for the same file), it can be done this way but we knowingly make this operation difficult so that people don't have many GUIDs to the same data by mistake.

Since only files on Storage Elements may be registered, `--destination-file` must refer to a Storage Element just like for `replicateFile`. The difference to `replicateFile` is that the latter requires the `sourceFileName` to be a valid grid file descriptor, i.e. either an LFN, GUID or other SURL, while `copyAndRegisterFile` does explicitly forbid such sources and accepts only non-grid files. For files already in the grid, `replicateFile` must be used.

The call returns the GUID of the new entry.

Arguments

`sourceFileName` The file to be registered. It must be accessible through this name using the specified protocol in the URI. The `sourceFileName` may be:

- A local file (i.e. a URI with `file` as its schema).
- A transport URI with a valid protocol schema (http, ftp, etc.).

Options

`destination-file` The physical destination file. `destination-file` may be specified in three different ways:

- As a fully qualified Storage URL (SURL). If this file already exists, the operation fails.
- Only the host of the SE where the file should be stored, i.e. the URI was constructed just by giving it a host string, which will be stored in the URI path. The storage location will be determined automatically depending on the user's VO and a suitable file name will be chosen.

- If no destination is specified at all (i.e. the option is omitted) the 'closest' SE is located and the file is stored as in the previous case.

`logical-file-name` The logical (file) name to be used in the catalog to find the file again. If this option is omitted, only the GUID returned by this call will be available to retrieve the file.

`protocol` The protocol to be used. If it is omitted, the default protocol for the given SE is used.

`streams` The number of parallel streams to be used for the copy. If omitted, the default is used for the given protocol.

Return value

The call returns 0 on success and -1 on failure. It also prints the GUID that this file was registered with on stdout.

Errors

The call can fail for many reasons. Some of the reasons are:

- The user has no/expired credentials.
- The `sourceFileName` does not exist.
- The `sourceFileName` or `destination-file` are invalid URIs.
- The `sourceFileName` or `destination-file` cannot be accessed.
- The copy operation fails.
- The given `logicalFileName` already exists.

3.1.2. `replicateFile`

Command:	**replicateFile**
Description:	Replicate a file to another SE.
Arguments:	**sourceFileName** The file to replicate. This may be an LFN, GUID or SURL. For LFN and GUID the best SURL is found through listBestFile. If the destination SE is omitted, the file is replicated to the local SE, i.e. it's the same as getBestFile.
Options:	**-d** or **- -destination** *destination-SURL* or *SE host* **-p** or **- -protocol** *protocol* **-n** or **- -streams** *number of streams*
Example:	`edg-replica-manager --vo=wpsix replicateFile lfn:mydata -d lxshare0384.cern.ch`

The help printed if the command is invoked without arguments or using `edg-rm -h replicateFile`:

```
usage: edg-replica-manager replicateFile sourceFileName [options]
Replicate a file to another SE.
 -d,--destination <file>    The destination to replicate to. This may be a
                            fully qualified SURL, just an SE host name or
                            just be omitted - in which case the replication
                            will be done to the closest SE.
 -n,--streams <#streams>    The number of parallel streams to use if
                            supported by protocol. Optional.
```

```
-p,--protocol <protocol>   The protocol to be used for the transfer
                           (optional).
Arguments:
 sourceFileName        The file to replicate. This may be an LFN, GUID or
                       SURL. For LFN and GUID the best SURL is found through
                       listBestFile. If the destionation SE is omitted,
                       the file is replicated to the local SE, i.e. it's
                       the same as getBestFile.
```

The `replicateFile` call performs the task of replicating a file between grid-aware stores and registering the replica in the Replica Catalog as an atomic operation. This is the preferred method to 'copy files around in the grid'.

The difference to `copyAndRegisterFile` is that this operation only allows for GUID, LFN or SURL as the source file whereas `copyAndRegisterFile` explicitly forbids that. No new GUID is generated for the replica (hence the term).

Semantics

`replicateFile` also handles all the possible failures if the registration should fail after the copy has taken place. If the remote site already contains a replica, this operation returns successfully very quickly.

Since replication can only be done between Storage Elements, both `sourceFileName` and `destination` must refer to a Storage Element (unless one uses LFN or GUID as the argument for the `sourceFileName` without further options).

The method returns the SURL of the new replica upon success returns with an appropriate error message upon failure.

Arguments

`sourceFileName` The file to be registered. The `sourceFileName` may be:

- A GUID. The SE will be chosen automatically.
- An LFN. The SE will be chosen automatically.
- An SURL. The given SE will be used as the source for this file.

Options

`destination-file` The physical destination may be specified in three different ways:

- As a fully qualified Storage URL (SURL). If this file already exists, the operation fails.
- Only the host of the SE where the file should be stored, i.e. the URI was constructed just by giving it a host string, which will be stored in the URI path. The storage location will be determined automatically depending on the user's VO and a suitable file name will be chosen.
- If no destination is specified at all (i.e. the option is omitted) the 'closest' SE is located and the SURL is determined as in the previous case.

`protocol` The protocol to be used. If it is omitted, the default protocol for the given SE is used.

`streams` The number of parallel streams to be used for the copy. If omitted, the default is used for the given protocol.

Return value

The call returns 0 on success and -1 on failure. It also prints the SURL of the new replica.

Errors

The call can fail for many reasons. Some of the reasons are:

- The user has no/expired credentials.
- The `sourceFileName` does not exist.
- The `sourceFileName` or `destination-file` are invalid URIs.
- The `sourceFileName` or `destination-file` cannot be accessed.
- The copy operation fails.

3.1.3. `deleteFile`

Command:	**deleteFile**
Description:	Delete a file from the Grid (and thus its Catalog). All replicas of a file may be deleted by specifying the GUID and setting the –all flag.
Arguments:	**fileName** File to delete. This has to be a grid-file, i.e. LFN, SURL or GUID. For GUIDs the –all flag may be set to delete all instances. For LFNs the storage option has to be set.
Options:	**-s** or **- -storage** *SE host* **-a** or **- -all**
Example:	`edg-replica-manager --vo=wpsix deleteFile guid:73e16e74-26b0-11d7-b1e0-c5c68d88236a --all`

The help printed if the command is invoked without arguments or using `edg-rm -h deleteFile`:

```
usage: edg-replica-manager deleteFile fileName [options]
Delete a file from the Grid. All replicas of a file may be deleted by
specifying the GUID and setting the --all flag.
 -s,--storage <host>   The storage host from which the physical instance is
                       to be deleted. This is mandatory for LFNs, ignored
                       for SURLs and mandatory for GUIDs if the --all flag
                       is not set.
 -a,--all              If this is set and the file-name is a GUID, delete
                       all mappings for the given GUID from the catalog and
                       remove all physical instances of the file.
 Arguments:
  fileName              File to delete. This has to be a grid-file, i.e.
                        LFN, SURL or GUID. For GUIDs the --all flag may be
                        set to delete all instances. For LFNs the storage
                        option has to be set.
```

The `deleteFile` call performs the task of removing a file from grid-aware storage and unregistering the entry in the Replica Catalog as an atomic operation.

Semantics

The `fileName` argument may be either a SURL, a LFN or a GUID. If it is an SURL, only the specified file will be deleted and the corresponding GUID-SURL mapping removed from the catalog. If the `fileName` is an LFN, the `storage` option is mandatory in order to specify which replica has to be removed. The same is true if the `fileName` argument is a GUID, but for GUIDs we allow to specify the `all` flag as well instead of the `storage option` which will remove all traces of the file from the Grid, including all LFN aliases to it.

Arguments

`fileName` The file to be deleted. The `fileName` may be:

- A GUID. The `storage` or `all` options are mandatory.
- An LFN. The `storage` option is mandatory, the `all` option is not allowed.
- An SURL. The given SURL will be removed.

Options

`storage` The host name of the SE where the file resides.

`all` A flag specifying whether all instances of the given GUID should be removed. An error is returned if this flag is given for an LFN or SURL.

Return value

The call returns 0 on success and -1 on failure.

Errors

The call can fail for many reasons. Some of the reasons are:

- The user has no/expired credentials.
- The `fileName` does not exist.
- The `fileName` is an invalid URI.
- The `storage` host is not an SE host.
- The delete operation fails or the user has no access rights.

3.2. CATALOG COMMANDS

The catalog commands are:

- `registerFile`
- `registerGUID`
- `unregisterFile`
- `listReplicas`
- `listGUID`
- `addAlias`
- `removeAlias`
- `printInfo`
- `getVersion`

3.2.1. registerFile

Command:	**registerFile**
Description:	Register a file that already is on a Grid-aware store. It returns the GUID with which the file was registered. Optionally an LFN may be given as well.
Arguments:	**SURL** The SURL of the file to register.
Options:	**-l** or **- -logical-file-name** *logical-file-name*
Example:	`edg-replica-manager --vo=wpsix registerFile srm://lxshare0384.cern.ch/flatfiles/alice/data/05/x.dat -l lfn:aliceprod/x`

The help printed if the command is invoked without arguments or using `edg-rm -h registerFile`:

```
usage: edg-replica-manager registerFile SURL [options]
Register a file that already is on a Grid-aware store. It returns the GUID
with which the file was registered. Optionally an LFN may be given as well.
 -l,--logical-file-name <lfn>   The logical file name to register this file
                                with
 Arguments:
  SURL                    The SURL of the file to register.
```

Register a file in the Replica Catalog that is already stored on a Storage Element. It takes two arguments: source file and logical name.

Semantics

The source file needs to be a qualified storage file name URI, following the rule

`srm://storage.element.host/path/file.name`

The logical name is optional; it may declare a logical identifier that can be used later to look up any instance of the file. The method returns the GUID, the Grid Unique IDentifier of the file. The details are:

- If the source file is not registered yet:
  - no logical name is given: a new GUID is generated, the file is registered and the GUID is returned to the caller.
  - a logical name is given: a new GUID will be allocated and returned, and the logical name will also be registered and can be used to retrieve the file later.
- If the file is already registered:
  - no logical name is given: the already existing GUID is returned.
  - a logical name is given that does not exist yet: it is added as an alias in the Replica Catalog so that the file can be looked up later using that name as well.
  - a logical name is given which already exists: it is checked whether the logical name really corresponds to the file. If not, an exception is thrown, otherwise the existing GUID is returned.

This method does not include any data movement (i.e. the storage file is not copied) and assumes that the storage file is already located at a known Storage Element. If the SURL does not exist on the SE, the command fails.

By specifying an SURL that is already registered, this method can be used to add a new alias to it (in addition to the `addAlias` command)

Arguments

`SURL` The file to be registered. It must be a valid Storage URL, i.e. the host part of the given URI needs to be a recognized Storage Element, as described above.

Options

`logical-file-name` The logical alias to be included in the catalog. Using this LFN the file may be retrieved as well, in addition to its GUID. If this option is omitted, the GUID will be the only name by which the file can be retrieved.

Return value

The call returns 0 on success and -1 on failure and prints the GUID with which this file was registered to stdout.

Errors

The call can fail for many reasons. Some of the reasons are:

- The user has no/expired credentials.
- The `SURL` does not exist.
- The `SURL or logical-file-name` is an invalid URI.
- The RLS or RMC are unattainable or ill-configured.

3.2.2. `registerGUID`

Command:	**registerGUID**
Description:	Register an SURL with a known GUID in the catalog
Arguments:	**SURL** The SURL to register. **GUID** The GUID to register the file with.
Options:	

Example:	`edg-replica-manager --vo=wpsix registerGUID`
	`srm://lxshare0384.cern.ch/flatfiles/alice/data/05/x.dat`
	`guid:73e16e74-26b0-11d7-b1e0-c5c68d88236a`

The help printed if the command is invoked without arguments or using `edg-rm -h registerGUID`:

```
usage: edg-replica-manager registerGUID SURL GUID
Register a file that is already in a Grid-aware store with a GUID given to
the command. This is only necessary if replication could not be carried out
through replicateFile for some reason, like the data was shipped by tape.

 Arguments:
  SURL                  The SURL of the file to register.
  GUID                  The known GUID of this copy.
```

Register a file in the Replica Catalog with a known GUID. The use case for this command is the so-called *truckFTP* use case where the replication happens not over the network but through a tape delivered by FedEx (maybe this is cheaper). The remote site wants to register the received data using a known GUID and declare the data a valid replica of an existing file.

Semantics

The command takes two arguments: source file and GUID. The source file needs to be a qualified storage file name URI, following the same rule for SURLs as for `registerFile` above. The GUID needs to be known to the system already. Be careful, if the GUID corresponds to a wrong file in the remote catalog the catalog data will become inconsistent. If the SURL does not exist on the SE, the command fails.

Arguments

`SURL` The file to be registered. It must be a valid SURL as described above.

`GUID` The GUID to register the file with.

Options

No options.

Return value

The call returns 0 on success and -1 on failure.

Errors

Some of the reasons for failure are:

- The user has no/expired credentials.
- The `SURL` does not exist.
- The `SURL` is an invalid URI.
- The `GUID` given is ill-formed or not a known GUID.
- The RLS or RMC are unattainable or ill-configured.

3.2.3. `unregisterFile`

Command:	**unregisterFile**
Description:	Unregister a file from the catalog.
Arguments:	**GUID** The GUID of the entry to unregister **SURL** The SURL to unregister
Options:	
Example:	`edg-replica-manager --vo=wpsix unregisterFile guid:1cc5353d-982f-11d7-9861-a4978670ee2b srm://lxshare0384.cern.ch/flatfiles/ali`

The help printed if the command is invoked without arguments or using `edg-rm -h unregisterFile`:

```
usage: edg-replica-manager unregisterFile GUID SURL
Unregister a file from the catalog.

 Arguments:
  GUID                  The GUID to unregister
  SURL                  The matching SURL to unregister
```

Unregister a file from the Replica Location Service that is stored on a Storage Element. It takes two arguments: the GUID and SURL to which the (GUID, SURL) mapping should be removed.

Note: in order to remove/unregister an LFN, use the command `removeAlias`.

Semantics

This command removes a replica of a file from 'grid awareness'. The file stored at the SE and identified with the SURL will not be removed (this is achieved using `deleteFile`. If the SURL does not actually exist on the SE, the command will be successful anyway. If the SURL specified is the only instance (replica) of the given file, the GUID will be removed as well, otherwise only the SURL is removed from the Catalog.

Note: Since LFNs (aliases) are treated like symbolic links in the Replica Metadata Catalog, if the last GUID is deleted from the RLS, the corresponding LFNs are *not* deleted from the Replica Metadata Catalog which results in "dangling LFNs" (similar to "dangling symbolic links").

```
srm://storage.element.host/path/file.name
```

Arguments

`GUID` The GUID to unregister for a given SURL.

`SURL` The file to be unregistered. It must be a valid Storage File Name, i.e. the host part of the given URI needs to be a recognized Storage Element, as described above.

Options

no options.

Return value

The call returns 0 on success and -1 on failure and prints the GUID with which this file was registered to stdout.

Errors

The call can fail for many reasons. Some of the reasons are:

- The user has no/expired credentials.
- The `SURL` does not exist.
- The `SURL or logical-file-name` is an invalid URI.
- The RLS or RMC are unattainable or ill-configured.

3.2.4. `listReplicas`

Command:	**listReplicas**
Description:	List all replicas of a file.
Arguments:	**fileName** The LFN, GUID or SURL to list all replicas of a file.
Options:	
Example:	`edg-replica-manager --vo=wpsix listReplicas lfn:mydat`

The help printed if the command is invoked without arguments or using `edg-rm -h listReplicas`:

```
usage: edg-replica-manager listReplicas fileName
List all replicas of a file.

 Arguments:
  fileName              The LFN, GUID or SURL to list all replicas of.
```

List replicas as they are registered in the Replica Catalog.

Semantics

Depending whether you specify an LFN, GUID or SURL as the input filename the edg-replica-manager will issue different commands to the underlying services. The result is always the same: a list of SURLs that are replicas of the same file.

- If a GUID is specified, then the RLS is contacted to resolve it into the corresponding list of SURLs, which is then returned and listed to stdout.
- If you specify an LFN as the filename argument, first the Replica Metadata Catalog is contacted to resolve the LFN into a proper GUID. Then the RLS is contacted to retrieve all corresponding registered SURLs.
- If an SURL is specified, the RLS is contacted to retrieve the corresponding GUID and then based on that GUID all other SURLs are found. So knowing an SURL you can find all available replicas. The SURL will be listed again in the output.

Arguments

`fileName` The file to get all replicas of. It must be a valid Logical File Name, GUID or Storage File Name.

Options

no options.

Return value

The call returns 0 on success and -1 on failure and prints all SURLs that are replicas of the input filename to stdout, one on each line.

Errors

The call can fail for many reasons. Some of the reasons are:

- The user has no/expired credentials.
- The `SURL, GUID or LFN` is invalid or does not exist.
- The RLS or RMC are unattainable or ill-configured.

3.2.5. `listGUID`

Command:	**listGUID**
Description:	Print the GUID associated with an LFN or SURL.
Arguments:	**LFNorSURL** The LFN or SURL to get the GUID of.
Options:	
Example:	`edg-replica-manager --vo=wpsix listGUID lfn:mydata`

The help printed if the command is invoked without arguments or using `edg-rm -h listGUID`:

```
usage: edg-replica-manager listGUID LFNorSURL
Get the GUID based on an LFN or SURL.

 Arguments:
  LFNorSURL              The LFN or SURL to get the GUID of.
```

Get the GUID that corresponds to a given SURL or LFN.

Semantics

Depending whether you specify an LFN or SURL as the input filename the edg-replica-manager will either access the Replica Metadata Catalog (in case of an LFN) to resolve the LFN into a GUID or the Replica Location Service (in case of an SURL) to find out what GUID the SURL is registered with. The command always returns a GUID.

Arguments

`LFNorSURL` The Logical or Site File Name. It must be a valid URI for LFN or SURL schemes.

Options

no options.

Return value

The call returns 0 on success (-1 on failure) and prints the GUID to stdout.

Errors

The call can fail for many reasons. Some of the reasons are:

- The user has no/expired credentials.
- The `SURL or LFN` is invalid or does not exist.
- The RLS or RMC are unattainable or ill-configured.

3.2.6. `addAlias`

Command:	**addAlias**
Description:	Add a new alias to GUID mapping
Arguments:	**GUID** The GUID to add the alias for **LFN** The LFN alias to add
Options:	
Example:	`edg-replica-manager --vo=wpsix addAlias guid:73e16e74-26b0-11d7-b1e0-c5c68d88236a lfn:important`

The help printed if the command is invoked without arguments or using `edg-rm -h addAlias`:

```
usage: edg-replica-manager addAlias GUID LFN
Add a new LFN alias to an existing GUID..

 Arguments:
  GUID                   The GUID to add the alias for.
  LFN                    The LFN alias to add.
```

Semantics

This command simply adds an LFN alias to an existing GUID. The LFN must be unique as well.

Arguments

`GUID` The GUID. It must be a valid GUID URI and must exist in the catalog.

`LFN` The new LFN. It must be a valid LFN URI and must be unique, i.e. not present in the catalog yet.

Options

no options.

Return value

The call returns 0 on success (-1 on failure).

Errors

The call can fail for many reasons. Some of the reasons are:

- The user has no/expired credentials.
- The `GUID or LFN` is invalid.
- The `GUID` does not exist.
- The `LFN` already exists.
- The RMC is unattainable or ill-configured.

3.2.7. `removeAlias`

Command:	**removeAlias**
Description:	Remove an alias LFN from a known GUID
Arguments:	**GUID** The GUID to remove the alias from **LFN** The LFN alias to add
Options:	
Example:	`edg-replica-manager --vo=wpsix removeAlias guid:73e16e74-26b0-11d7-b1e0-c5c68d88236a lfn:important`

The help printed if the command is invoked without arguments or using `edg-rm -h removeAlias`:

```
usage: edg-replica-manager removeAlias GUID LFN
Remove an LFN alias to GUID mapping..

 Arguments:
  GUID                  The GUID to remove the alias of.
  LFN                   The LFN alias to remove.
```

Semantics

This command removes an LFN alias from an existing GUID. The LFN should be unique as well.

Arguments

`GUID` The GUID. It must be a valid GUID URI and must exist in the catalog.

`LFN` The LFN. It must be a valid LFN URI and must exist in the catalog.

Options

no options.

Return value

The call returns 0 on success (-1 on failure).

Errors

The call can fail for many reasons. Some of the reasons are:

- The user has no/expired credentials.
- The `GUID or LFN` is invalid.
- The `GUID` does not exist.
- The `LFN` does not exist.
- The RMC is unattainable or ill-configured.

3.2.8. `printInfo`

Command:	**printInfo**
Description:	Print the information needed by the Replica Manager to screen or to a file.
Arguments:	none
Options:	**-f** or **–file** *File to print the info to.*
Example:	`edg-replica-manager --vo=wpsix printInfo`

The help printed if the command is invoked without arguments or using `edg-rm -h printInfo`:

```
usage: edg-replica-manager printInfo [options]
Print the information needed by the Replica Manager to screen or to a file.
 -f,--file <file>   The file to print the info to. This file can be used as
                    a properties file for the Stub Info system.
```

Print information about:

- replication services (RMC, LRC, ROS) and their URLs
- Information Service used (R-GMA, MDS, local configuration file)
- Storage Elements and Computing Elements with some of their attributes

This command can be used for debugging as well as for retrieving information on which storage resources should be accessible by the replica manager.

3.2.9. getVersion

Command:	**getVersion**
Arguments:	none.
Options:	**-h** *hostname* **-p** *port*
Example:	`edg-replica-manager getVersion -h localhost`
Return Values:	Client version = 1.4.3

The help printed if the command is invoked using `edg-rm getVersion -h`:

```
usage: edg-replica-manager getVersion
Retrieve the version of both the server and the client.
```

Get the versions of the replica manager client.

3.3. Optimization Commands

The catalog commands are:

- `listBestFile`
- `getBestfile`
- `getAccessCost`

3.3.1. listBestFile

Command:	**listBestFile**
Description:	Return the SURL that has the smallest access cost from the local store (or from the store specified by the -d option).
Arguments:	**LFNorGUID** The LFN or GUID for which we want to list the 'best' SURL as seen from the destination specified by the -d option (or the local SE if omitted)
Options:	**-d** or **- -destination** *SE host*
Example:	`edg-replica-manager --vo=wpsix listBestFile lfn:important`

The help printed if the command is invoked without arguments or using `edg-rm -h listBestFile`:

```
usage: edg-replica-manager listBestFile LFNorGUID [options]
Return the SURL that has the smallest access cost from the local store (or
from the store specified by the -d option).
 -d,--destination <host>   The destination SE. The file to be found is the
                           best source for a copy to this destination. If
                           it's omitted, the local SE is used.
 Arguments:
  LFNorGUID             The LFN or GUID for which we want to list the
                        'best' SURL as seen from the destination specified
                        by the -d option (or the local SE if omitted)
```

List the best replica.

Semantics

First the LFN or GUID specified is resolved into a list of SURLs as described in the `listReplicas` command above. Then the Replica Optimization Service is contacted with the given list of SURLs, and the specified destination host (or the local SE host if none was specified). The ROS will return the SURL that has the smallest access cost from the given host at this point in time.

Arguments

`LFNorGUID` The LFN or GUID of the file of which the 'best' replica should be found.

Options

`destination` The host that is the point of reference for calculating the network and storage access cost. If not specified, the localSE is used as found through the information providers.

Return value

The call returns 0 on success and -1 on failure. It prints the 'best' SURL on success to stdout.

Errors

The call can fail for many reasons. Some of the reasons are:

- The user has no/expired credentials.
- The `GUID` or `LFN` is invalid or does not exist.
- The RLS or RMC or ROS are unattainable or ill-configured.

3.3.2. `getBestFile`

Command:	**getBestFile**
Description:	Make a file available on local storage (or on the store specified by the -d option).
Arguments:	**LFNorGUID** The LFN or GUID for which we request a copy to be present at the destination specified by the -d option (or the local SE if omitted)
Options:	**-d** or **- -destination** *SE host* **-p** or **- -protocol** *protocol* **-n** or **- -streams** *number of streams*
Example:	`edg-replica-manager --vo=wpsix getBestFile lfn:important`

The help printed if the command is invoked without arguments or using `edg-rm -h getBestFile`:

```
usage: edg-replica-manager getBestFile LFNorGUID [options]
Make a file available on local storage (or on the store specified by the -d
option).
 -d,--destination <host>    The destination SE. The file to be found is the
                            best source for a copy to this destination. If
                            it's omitted, the local SE is used.
 -n,--streams <#streams>    The number of parallel streams to use. Defaults
                            to 8.
 -p,--protocol <protocol>   The protocol to be used for an eventual
```

```
                    transfer. Defaults to gsiftp.
Arguments:
 LFNorGUID          The LFN or GUID for which we request a copy to be
                    present at the destination specified by the -d
                    option (or the local SE if omitted)
```

Make the file available from local storage (or the storage specified by the destination option). The replica manager will find the best source to copy the file from, or will exit immediately if the file is already available on the given storage.

Semantics

First the LFN or GUID specified is resolved into a list of SURLs as described in the `listReplicas` command above. Then it is checked whether one of the replicas is actually on local storage (or the destination storage) in which case the command lists that file and exits.

If the file is not available, a listBestFile operation is carried out, with the given destination, and the best source for replication is found. Then that file is replicated to the destination just like described in the `replicateFile` command's section.

Arguments

`LFNorGUID` The LFN or GUID of the file for which we need a local replica (or on the specified destination).

Options

`destination` The destination to replicate to. If not specified, the localSE is used as found through the information providers.

`protocol` The protocol to be used. If it is omitted, the default protocol for the given SE is used.

`streams` The number of parallel streams to be used for the copy. If omitted, the default is used for the given protocol.

Return value

The call returns 0 on success and -1 on failure. It prints the SURL that can be used locally (or on the specified destination) upon success to stdout.

Errors

The call can fail for many reasons. Some of the reasons are:

- The user has no/expired credentials.
- The `GUID or LFN` is invalid or does not exist.
- The RLS or RMC or ROS are unattainable or ill-configured.

3.3.3. getAccessCost

Command:	**getAccessCost**
Description:	Get the access cost of the cheapest replicas for each LFN at each CE site.
Arguments:	
Options:	**-d** or **- -destination** *list of CE hosts* **-l** or **- -logical-file-name** *list of logical-file-names* **-p** or **- -protocol** *protocol*
Example:	`edg-replica-manager --vo=wpsix getAccessCost -l lfn:one lfn:two -d lxshare0384.cern.ch grid01.nikhef.nl`

The help printed if the command is invoked without arguments or using `edg-rm -h getAccessCost`:

```
usage: edg-replica-manager getAccessCost [options]
Get the access cost of the cheapest replicas for each LFN at each CE site.
 -d,--computing-elements <ce>     The destination Computing Elements.
 -l,--logical-file-names <file>   The logical file names
 -p,--protocol <protocol>         The protocol to be used for transfer.
                                  Defaults to gsiftp.
```

Calculate the expected file access cost per-Logical-File-Name. It makes the assumptions that no replication occurs. The `-l` and `-d` options are mandatory.

Semantics

First the list of LFNs is resolved into a corresponding list of lists of SURLs as described in the `listReplicas` command above. The list of CEs is extended with their corresponding localSE, as found from the information provider.

All this information is submitted to the Replica Optimization Service that will call the network monitors and calculate the overall lowest possible access cost for all LFNs at the given CEs.

The access-cost array is returned, which is then printed on screen, with the best SURL for each LFN at each CE, with the corresponding estimated time to actually copy the file to local store.

Arguments

no arguments.

Options

`computing-elements` A list of valid CE hostnames.

`logical-file-names` A list of valid LFNs or GUIDs.

`protocol` The protocol to be used. If it is omitted, the default protocol used, currently GridFTP.

Return value

The call returns 0 on success and -1 on failure. It prints the access cost to stdout, grouped by CEs and sub-grouped by LFNs.

Errors

The call can fail for many reasons. Some of the reasons are:

- The user has no/expired credentials.
- The `GUID or LFN` is invalid or does not exist.
- The RLS or RMC or ROS are unattainable or ill-configured.

3.4. FILE TRANSFER COMMANDS

The file transfer commands are:

- `copyFile`

3.4.1. COPYFILE

Command:	**copyFile**
Description:	Copy a file to a non-grid destination.
Arguments:	**sourceFileName** The file to copy from. This can be any kind of file: LFN, GUID, SURL, TURL or local file. **destFileName** The destination. This can only be a non-grid file - i.e. local file or TURL.
Options:	**-p** or **- -protocol** *protocol* **-n** or **- -streams** *number of streams*
Example:	`edg-replica-manager --vo=wpsix copyFile lfn:mydata file:/home/bob/analysis/data5.dat`

The help printed if the command is invoked without arguments or using `edg-rm -h copyFile`:

```
usage: edg-replica-manager copyFile sourceFileName destFileName [options]
Copy a file to a non-grid destination.
 -f,--force                 Overwrite the destination if it's already
                            there.
 -n,--streams <#streams>    The number of parallel streams to use. Defaults
                            to 8.
 -p,--protocol <protocol>   The protocol to be used for transfer. Defaults
                            to gsiftp.
 Arguments:
  sourceFileName        The file to copy from. This can be any kind of
                        file: LFN, GUID, SURL, TURL or local file.
  destFileName          The destination. This can only be a non-grid file -
                        i.e. local file or TURL.
```

Copies a physical file from `source` to `destination` using the specified transport mechanism.

Semantics

This action does not involve any updates to the replica catalogs as the destination cannot be a grid-aware store, in order to avoid catalog corruption. copyAndRegisterFile needs to be used for that purpose, i.e. to bring a file into the grid. The destination does not accept GUIDs, LFNs or SURLs. As said, for those cases copyAndRegisterFile or replicateFile needs to be used.

It provides all the capability of globus-url-copy and more, accepting also GUIDs, LFNs and SURLs as the source file. Since the destination is not on grid-storage, the destination file will not be registered in the catalog.

Example: To get a local copy of a grid file, the source file can be specified as a GUID or LFN and the destination file as a local file. The result will be that the file will be copied 'out of the grid' to the local file. Local file URIs need to have the 'file' scheme.

Arguments

`sourceFileName` The source file. It may be one of the following:

- A GUID - the 'best' SE will be located to find the cheapest replica to copy from
- An LFN - the 'best' replica will be located as for the GUID
- A valid SURL, having the SE as its host name
- A valid transport URI, with a real protocol as its schema. Currently http, https, ftp and gsiftp are supported.
- A local file (specified with the 'file' schema).

`destFileName` The physical destination file. Must be one of the following:

- A local file
- A transport URI with a valid protocol. The same protocols are supported as for the source.

Options

`protocol` The protocol to be used. If it is omitted, the default protocol for the given SE is used. This parameter is only considered if the source is specified using an SURL, LFN or GUID.

`streams` The number of parallel streams to be used for the copy. If omitted, the default is used for the given protocol.

Return value

The call returns 0 on success and -1 on failure.

Errors

The call can fail for many reasons. Some of the reasons are:

- The user has no/expired credentials.
- The `source URI` is invalid or does not exist.
- The `destination URI` is invalid or not writable.
- The RLS or RMC or ROS are unattainable or ill-configured.

4. QUICK START AND SIMPLE HOW TO

In this section we give a few practical examples that can be directly applied to the EDG testbed. We assume that the user has a valid Grid proxy certificate.

Since all replication tools use the Grid security infrastructure, we first need to make sure that we have a valid user proxy. To check this, use the following command:

```
[hst@testbed010] grid-proxy-info -all
subject  : /O=Grid/O=CERN/OU=cern.ch/CN=Heinz Stockinger/CN=proxy
issuer   : /O=Grid/O=CERN/OU=cern.ch/CN=Heinz Stockinger
type     : full
strength : 512 bits
timeleft : 11:59:45
```

In case the user proxy is not available or has expired, renew it by issuing

```
[hst@testbed010] grid-proxy-init
Your identity: /O=Grid/O=CERN/OU=cern.ch/CN=Heinz Stockinger
Enter GRID pass phrase for this identity:
Creating proxy ............................................. Done
Your proxy is valid until Sat Oct 19 04:44:30 2002
```

4.1. First Steps

In order to get familiar with the way the command line tool has to be used, you can try the basic replica manager command to retrieve the version of the client program you are using. The output of the command is given, too.

```
[hst@lxshare0313] edg-rm getVersion
Client version = 1.4.3
```

As a next step, you might want to get information about the resources in the testbed that are available to you. Here, we also remind you that you need to specify your VO (here, we assume wpsix) in the order as given below. We do not list the entire output of the command.

```
[hst@lxshare0313] edg-rm --vo=wpsix printInfo
VO used            : wpsix
default SE         : pcrd24.cern.ch
default CE         : lxshare0313.cern.ch
Info Service class : org.edg.data.reptor.info.InfoServiceStub

RMC endpoint       : http://lxshare0342.cern.ch:8080/edg-replica-metadata-catalog/services/edg-replica-metadata-catalog
LRC endpoint       : http://lxshare0344.cern.ch:8080/edg-replica-location/services/edg-local-replica-catalog
ROS endpoint       :
http://lxshare0343.cern.ch:8080/edg-replica-optimization/services/edg-replica-optimization

List of CE ID's     : lxshare0313.cern.ch
                      ccgridli01.in2p3.fr
                      gppce06.gridpp.rl.ac.uk
                      ce01.nikhef.nl
                      testbed001.cnaf.infn.it
                      grid01.ph.gla.ac.uk
[...]
SE at RAL :
                       name : RAL
                       host : gppse06.gridpp.rl.ac.uk
                       type : disk
                        VOs : wpsix
               VO Directory : wpsix : /flatfiles/06/wpsix
                  protocols : gsiftp
```

4.2. Replication Examples using Optimization Functions

This example runs through a scenario how the edg-replica-manager commands might be used to achieve standard data management tasks. In the scenario the user knows that there is a file available at CERN,

that has been put on a host accessible through GridFTP. It is not a grid-aware store, so first the user has to copy the file to a Storage Element and register it in the Grid. Say that for some reason the user cannot copy it to the local CERN Storage Element but has to copy it to the one at IN2P3.

In the example the file is called 'higgs0' and resides at `testbed008.cern.ch/tmp/`.

The copy and registration is an atomic operation. In the example we assign also a Logical File Name alias to it in the process, lfn:higgs, which is easier to remember than the GUID that is returned by the call:

```
edg-rm --vo=wpsix copyAndRegisterFile gsiftp://testbed008.cern.ch/tmp/higgs0
-l lfn:higgs
-d srm://ccgridli02.in2p3.fr/edg/StorageElement/dev2/wpsix/higgs
```

A GUID is created and returned to the screen:

```
guid:7c29f32b-4964-11d7-a86c-9ee9a33b1f19
```

To verify whether the operation got successfully executed, we can issue listReplicas:

```
edg-rm --vo=wpsix listReplicas lfn:higgs
```

which yields:

```
srm://ccgridli02.in2p3.fr/edg/StorageElement/dev2/wpsix/higgs
```

In order to retrieve the GUID based on the LFN, we can issue

```
edg-rm --vo=wpsix listGUID lfn:higgs
```

As a second step, the user might want to have a replica of this data file available at NIKHEF, because he intends to share it or to submit jobs that require resources at NIKHEF. A replica can be created using the replicateFile command:

```
edg-rm --vo=wpsix replicateFile lfn:higgs -d se01.nikhef.nl/flatfiles/wpsix/higgs
```

The command confirms its execution by returning the actual SURL used. If the -d option is omitted, an automatic SURL would have been created. Here the output is:

```
srm://se01.nikhef.nl/flatfiles/wpsix/higgs
```

To list all replicas now in the system, we can issue listReplicas again:

```
edg-rm --vo=wpsix listReplicas lfn:higgs
```

which yields:

```
srm://ccgridli02.in2p3.fr/edg/StorageElement/dev2/wpsix/higgs
srm://se01.nikhef.nl/flatfiles/wpsix/higgs
```

To see which replica has the best network connection to CERN, we can use listBestFile:

```
edg-rm --vo=wpsix listBestFile lfn:higgs -d pcrd24.cern.ch
```

The output is, for example:

```
srm://se01.nikhef.nl/flatfiles/wpsix/higgs
```

which means that the file at NIKHEF can be made available at CERN faster than the one from LYON. We now want to see the file access costs of the best replica with respect to CERN, NIKHEF and Lyon.

```
edg-rm --vo=wpsix getAccessCost -l lfn:higgs -d lxshare0313.cern.ch ce01.nikhef.nl ccgridli01.
```

The output is:

```
Access Cost 0 :
CE = lxshare0313.cern.ch
0:srm://se01.nikhef.nl/flatfiles/wpsix/higgs
TotalTime = 0.13

Access Cost 1 :
CE = ce01.nikhef.nl
0:srm://se01.nikhef.nl/flatfiles/wpsix/higgs
TotalTime = 0.0

Access Cost 2 :
CE = ccgridli01.in2p3.fr
0:srm://ccgridli02.in2p3.fr/edg/StorageElement/dev2/wpsix/higgs
TotalTime = 0.0
```

The list is grouped by the CEs given on the command line. For each CE the 'best' replica is listed and the time it would take to make it available locally. We can see that the expected access cost to CERN is 0.13 sec for the replica from NIKHEF, which is better than the one at Lyon (so that one is not listed at all). For the other sites the access cost is 0 since the file is already locally available and no network transfer is required.

To actually make the best file available at CERN, we can issue getBestFile

```
edg-rm --vo=wpsix getBestFile lfn:higgs -d pcrd24.cern.ch
```

The output is something like:

```
srm://pcrd24.cern.ch/data/temp/a6289c7c-4966-11d7-bc63-d91230733e2d
```

We should now have three replicas:

```
edg-rm --vo=wpsix listReplicas lfn:higgs
```

The output is:

```
srm://pcrd24.cern.ch/data/temp/aaa64014-4967-11d7-a6cc-f7a1ff1899b0
srm://se01.nikhef.nl/flatfiles/wpsix/higgs
srm://ccgridli02.in2p3.fr/edg/StorageElement/dev2/wpsix/higgs
```

To delete a replica we can use the deleteFile command:

```
edg-rm deleteFile lfn:higgs -s ccgridli02.in2p3.fr
```

5. APPENDIX: EDG REPLICA MANAGER LOG FILE

The edg-replica-manager commands are logged in a file called edg-replica-manager-UID.log which by default is written into the directory /tmp if logging is switched on. UID corresponds to your Unix user ID.

It contains logging information depending on the contents of the

```
$EDG_LOCATION/etc/edg-replica-manager/log4j.properties
```

file. Depending on the logging level you get more or less information from the different packages using log4j.

By default, logging is turned off and set as follows in the file `log4j.properties`:

```
# all logging is turned OFF by default.
log4j.rootLogger=OFF, FILE
```

You need to set this parameter to “ON” as follows:

```
log4j.rootLogger=ON, FILE
```

To increase the default logging level (INFO), change the log level to DEBUG on the line

```
log4j.logger.org.edg.data=DEBUG
```

Of course users normally do not have write access to the log4j properties file in the $EDG_LOCATION, so in order to have individual logging properties, save a copy of this file in a writable directory and save your changes there. In order for edg-replica-manager to recognize your file, say /tmp/mylog4j.props, set your REPTOR_OPTS environment variable to

```
export REPTOR_OPTS=-Dlog4j.configuration=file:/tmp/mylog4j.props
```

Upon subsequent runs of edg-replica-manager, the new logger settings will take effect. For more customization options in the log4j properties file, we refer to the log4j user manual [8].

As an example, we assume a UID of `testuser` and logging switched on. In this case, you find the log file at:

```
[testuser@lxshare0409] more /tmp/edg-replica-manager-testuser.log
2003-06-13 10:55:39,838 DEBUG [main] util.CommandFactoryImpl  - readConfig() : Loading Config
2003-06-13 10:55:39,941 DEBUG [main] util.CommandClient  - addOption : shortopt = i
2003-06-13 10:55:39,942 DEBUG [main] util.CommandClient  - addOption : shortopt =
2003-06-13 10:55:39,942 DEBUG [main] util.CommandClient  - addOption : shortopt
```

6. APPENDIX: REPTOR VERSUS EDG-REPLICA-MANAGER

The `edg-replica-manager` (first release in May 2002) can be regarded as a prototype replica manager for Reptor, the new EDG Replica Manager. In this section we describe a few differences between the previous `edg-replica-manager` and Reptor in order to provide an easier start for people already familiar with `edg-replica-manager`. Whenever we use the program name `edg-replica-manager` we refer to the old tool, and we use the term Reptor or EDG Replica Manager to refer to the new replication tool.

Basically, Reptor provides a very similar interface like the `edg-replica-manager`. However, there are more features added and more Grid services used. The following list outlines the main differences.

- Reptor uses the **Replica Location Service (RLS)** and the **Replica Metadata Catalog** and thus there is no restriction of the use of Logical File Names. The LFN can be any user supplied string prefixed by `lfn:`. In addition, alias names are allowed for LFNs.
- In addition to LFNs, **GUIDs (Grid Universal Identifiers)** are uses to uniquely identify Logical File Names.
- For all physical filenames or site filenames, a **protocol needs to be added as a prefix**. For `edg-replica-manager` the hostname alone was sufficient but Reptor also requires the protocol name in the file name. Example:

  `edg-replica-manager` (old):

  ```
  copyAndRegisterFile -s host1.cern.ch/home/data/testfile
  ```

  Reptor:

  ```
  copyAndRegisterFile gsiftp://host1.cern.ch/home/data/testfile
  ```

  Also the protocol ”file:” is allowed if a file is available locally.
- The command line interface (CLI) has changed slightly: instead of having an executable for each method (edg-replica-manager), Reptor provides a single executable where the method is passed as the first parameter.

  ```
  edg-rm copyAndRegister
  ```

  rather than

  ```
  edg-replica-manager-copyAndRegister
  ```

  Reptor has one **Java** rather than several C++ executables.
- No direct MSS interface: EDG Replica Manager/Reptor uses the SRM for that purpose
- The EDG Replica Manager package is a pure Java package and does not provide a C++ interface to the end user. A limited C++ API is provided for the methods listReplicas and getAccessCosts and is part of the edg-replica-manager-client package (C++ based).
- Reptor uses the Replica Optimization Services (ROS, also called Optor) and thus provides several replica optization methods like getBestFile, listBestFile, getAccessCost etc.
- registerFile/unregisterFile instead of registerEntry/unregisterEntry

REFERENCES

[1] Leanne Guy, Erwin Laure, Peter Kunszt, Heinz Stockinger, Kurt Stockinger. Replica Management in Data Grids. Technical report, Global Grid Forum Informational Document, GGF5, Edinburgh, Scotland, July 2002.
http://edms.cern.ch/document/350430

[2] WP2 DataManagement, WP2 Replica Manager Design Specification, Draft 0.7, 21 March 2002.
http://edms.cern.ch/document/338668/0.7

[3] Ann Chervenak, Ewa Deelman, Ian Foster, Wolfgang Hoschek, Adriana Iamnitchi, Carl Kesselman, Peter Kunszt, Matei Ripeanu, Heinz Stockinger,Kurt Stockinger, and Brian Tierney. Giggle: A Framework for ConstructingScalable Replica Location Services. In Proc. of the Int'l. IEEE Supercomputing Conference (SC 2002) (to appear), Baltimore, USA, November 2002.

[4] RLS: http://cern.ch/grid-data-management/replica-location-service

[5] edg-replica-manager: http://cern.ch/grid-data-management/edg-replica-manager

[6] R-GMA Documentation: http://hepunx.rl.ac.uk/edg/wp3/documentation/index.html

[7] W. H. Bell, D. G. Cameron, L. Capozza, P. Millar, K. Stockinger, F. Zini. Design of a Replica Optimisation Framework. Technical Report DataGrid-02-TED-021215, CERN, Geneva, Switzerland, December 2002.

[8] Ceki Gülcü. Short introduction to log4j. http://jakarta.apache.org/log4j/docs/manual.html